

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



MÔN HỌC: VI XỬ LÝ - VI ĐIỀU KHIỂN (TN) (CO3010)

LAB 3 - BUTTONS AND SWITCHES

Giảng viên hướng dẫn: Lê Trọng Nhân
Tôn Huỳnh Long

Lớp: L02

Sinh viên thực hiện: Võ Thị Ánh Tuyết - 2213826



Contents

1	Exercise 1: Sketch an FSM	2
2	Exercise 2: Proteus Schematic	2
3	Exercise 3: Create STM32 Project	3
4	Exercise 4: Modify Timer Parameters	6
5	Exercise 5: Adding code for button debouncing	6
6	Exercise 6: Adding code for displaying modes	10
7	Exercise 7: Adding code for increasing time duration value for the red LEDs, amber, greenLEDs	16
8	Link	22

1 Exercise 1: Sketch an FSM

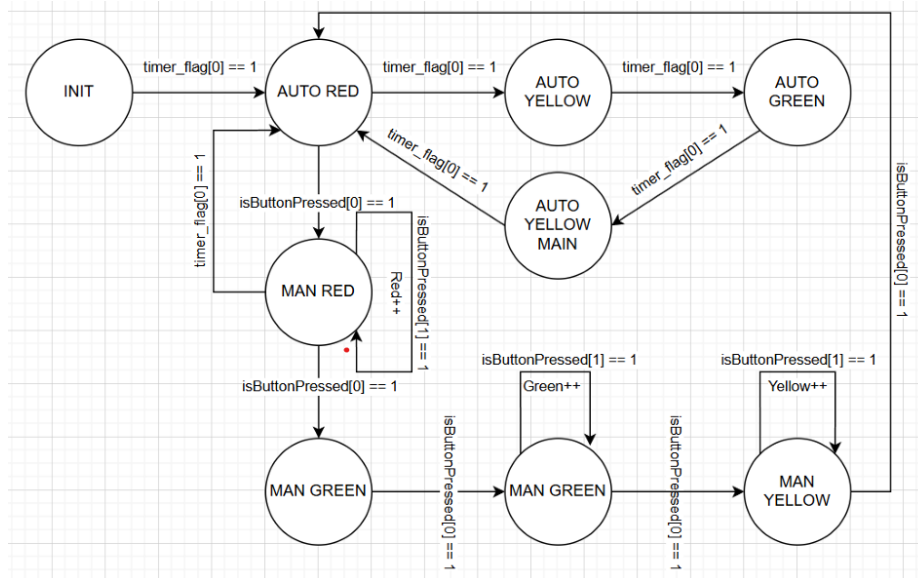


Figure 1: Exercise 1

2 Exercise 2: Proteus Schematic

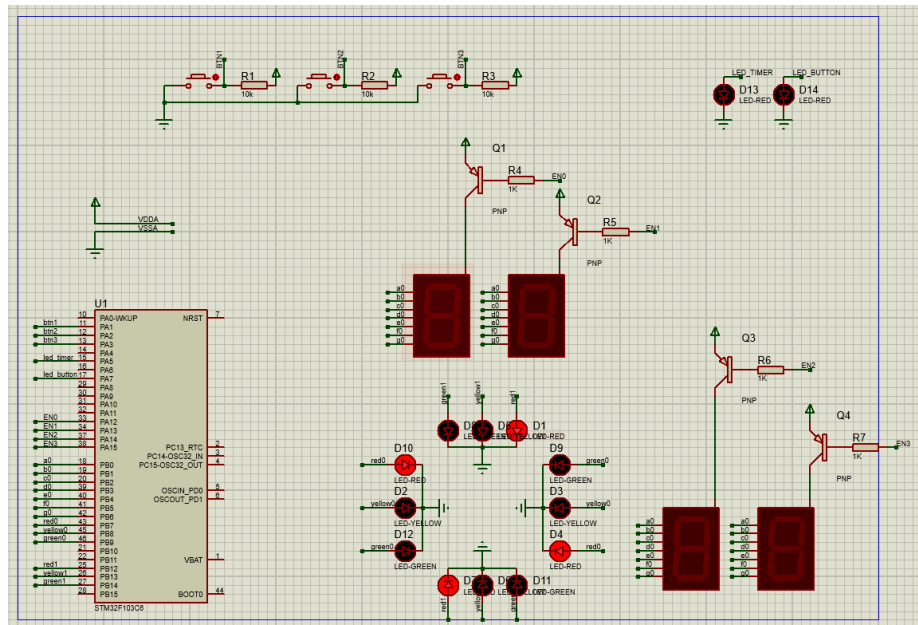


Figure 2: Exercise 2

3 Exercise 3: Create STM32 Project

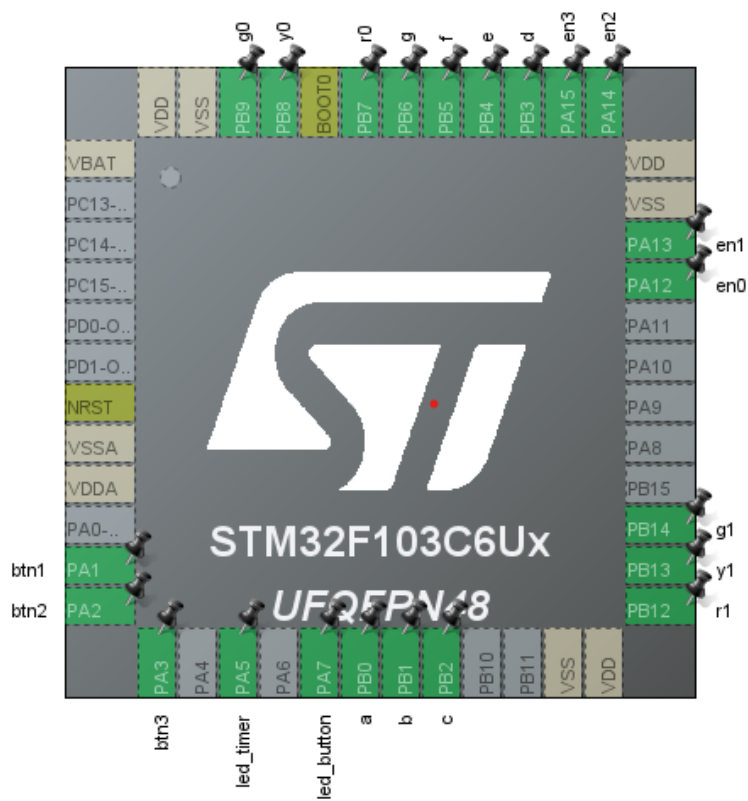


Figure 3: Exercise 3

```
1 #include "software_timer.h"
2 int timer_flag[100];
3 int timer_counter[100];
4
5 void set_timer(int index, int counter)
6 {
7     timer_flag[index] = 0;
8     timer_counter[index] = counter/tick;
9 }
10
11 void led_timer()
12 {
13     HAL_GPIO_TogglePin(GPIOA, GPIO_Pin_11);
14 }
15
16 void timer_run()
17 {
18     for(int i = 0; i < 10; i++)
19     {
20         if(timer_counter[i] >= 0)
21         {
22             timer_counter[i]--;
23             if(timer_counter[i] <= 0)
24             {
25                 timer_flag[i] = 1;
26             }
27         }
28     }
29 }
30
31 int is_timer_expired(int index)
32 {
33     if(index < 0 || index >= MAX_TIMERS)
34     {
35         return -1;
36     }
37     return timer_flag[index];
38 }
```

Listing 1: Source code in the software_timer.c

4 Exercise 4: Modify Timer Parameters

I use the variable *tick* to represent the timer interrupt period.
When setting the duration for the software timer, I divide *duration_ms* by *tick* to calculate the number of required interrupts.
Each time the timer interrupt occurs, the function *timer_run()* decreases the counters. When a counter reaches zero, the variable *timer_flag* is set.
With this method, even if the timer interrupt period (*tick* = 1 ms, 10 ms, or 100 ms) changes, the total real-time duration of each timer remains accurate, and the LED continues to blink at exactly 2 Hz.

5 Exercise 5: Adding code for button debouncing

```
1 #include "button.h"
2 int KeyReg00 = NORMAL_STATE;
3 int KeyReg10 = NORMAL_STATE;
4 int KeyReg20 = NORMAL_STATE;
5 int KeyReg30 = NORMAL_STATE;
6 int KeyReg01 = NORMAL_STATE;
7 int KeyReg11 = NORMAL_STATE;
8 int KeyReg21 = NORMAL_STATE;
9 int KeyReg31 = NORMAL_STATE;
10 int KeyReg02 = NORMAL_STATE;
11 int KeyReg12 = NORMAL_STATE;
12 int KeyReg22 = NORMAL_STATE;
13 int KeyReg32 = NORMAL_STATE;
14
15 int TimeOutForKeyPress = 50;
16 int button0_pressed = 0;
17 int button1_long_pressed = 0;
18 int button0_flag = 0;
19 int button1_pressed = 0;
20 int button1_flag = 0;
21 int button2_pressed = 0;
22 int button2_flag = 0;
23
24 int isButton0Pressed(){
```

```
25     if(button0_flag == 1){
26         button0_flag = 0;
27         return 1;
28     }
29     return 0;
30 }
31 int isButton1Pressed(){
32     if(button1_flag == 1){
33         button1_flag = 0;
34         return 1;
35     }
36     return 0;
37 }
38 int isButton2Pressed(){
39     if(button2_flag == 1){
40         button2_flag = 0;
41         return 1;
42     }
43     return 0;
44 }
45
46 int isButton1LongPressed(){
47     if(button1_long_pressed == 1){
48         button1_long_pressed = 0;
49         return 1;
50     }
51     return 0;
52 }
53
54
55
56 void getKeyInput()
57 {
58     KeyReg20 = KeyReg10;
59     KeyReg10 = KeyReg00;
60     // Add your key
61     KeyReg00 = HAL_GPIO_ReadPin(btn1_GPIO_Port, btn1_Pin);
62
63     if ((KeyReg10 == KeyReg00) && (KeyReg10 == KeyReg20))
```



```
64 {
65     if (KeyReg20 != KeyReg30)
66     {
67         KeyReg30 = KeyReg20;
68         if (KeyReg30 == PRESSED_STATE)
69         {
70             TimeOutForKeyPress = 500;
71             //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
72             button0_flag = 1;
73         }
74     }
75 }
76 else
77 {
78     TimeOutForKeyPress --;
79     if (TimeOutForKeyPress == 0)
80     {
81         TimeOutForKeyPress = 500;
82         if (KeyReg30 == PRESSED_STATE)
83         {
84             //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
85             button0_flag = 1;
86         }
87     }
88 }
89 }
90 KeyReg21 = KeyReg11;
91 KeyReg11 = KeyReg01;
92 // Add your key
93 KeyReg01 = HAL_GPIO_ReadPin(btn2_GPIO_Port, btn2_Pin);
94
95 if ((KeyReg11 == KeyReg01) && (KeyReg11 == KeyReg21))
96 {
97     if (KeyReg21 != KeyReg31)
98     {
99         KeyReg31 = KeyReg21;
100         if (KeyReg31 == PRESSED_STATE)
101         {
102             TimeOutForKeyPress = 500;
```

```
103         //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
104         button1_flag = 1;
105     }
106 }
107 else
108 {
109     TimeOutForKeyPress --;
110     if (TimeOutForKeyPress == 0)
111     {
112         TimeOutForKeyPress = 500;
113         if (KeyReg31 == PRESSED_STATE)
114         {
115             //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
116             button1_flag = 1;
117         }
118     }
119 }
120 }
121 KeyReg22 = KeyReg12;
122 KeyReg12 = KeyReg02;
123 // Add your key
124 KeyReg02 = HAL_GPIO_ReadPin(btn3_GPIO_Port, btn3_Pin);
125
126 if ((KeyReg12 == KeyReg02) && (KeyReg12 == KeyReg22))
127 {
128     if (KeyReg22 != KeyReg32)
129     {
130         KeyReg32 = KeyReg22;
131         if (KeyReg32 == PRESSED_STATE)
132         {
133             TimeOutForKeyPress = 500;
134             //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
135             button2_flag = 1;
136         }
137     }
138     else
139     {
140         TimeOutForKeyPress --;
141         if (TimeOutForKeyPress == 0)
```

```
142         {
143             TimeOutForKeyPress = 500;
144             if (KeyReg32 == PRESSED_STATE)
145             {
146                 //HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
147                 button2_flag = 1;
148             }
149         }
150     }
151 }
152 }
```

Listing 2: Source code button.c

6 Exercise 6: Adding code for displaying modes

To add code for display mode on seven-segment LEDs

```
1 #include "fsm_auto.h"
2
3 int i = 0;
4 int led_buffer[4] = {0, 0, 0, 0};
5 int num_to_display_1 = 0;
6 int num_to_display_2 = 0;
7
8 void fsm_auto()
9 {
10     if(is_timer_expired(3) && (status == AUTO_GREEN_RED ||
11                               status == AUTO_RED_GREEN ||
12                               status == AUTO_YELLOW_RED ||
13                               status == AUTO_RED_YELLOW ||
14                               status == AUTO_INIT))
15     {
16         getKeyInput();
17         if (isButton0Pressed())
18         {
19             HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_7);
20             clear();
21             led_buffer[0] = 0;
22             led_buffer[1] = 0;
```

```
23     led_buffer[2] = 0;
24     led_buffer[3] = 0;
25     status = MANUAL_INIT;
26 }
27 else set_timer(3, 20);
28 }
29
30 if(is_timer_expired(5)  && (status == AUTO_GREEN_RED ||
31                             status == AUTO_RED_GREEN ||
32                             status == AUTO_YELLOW_RED ||
33                             status == AUTO_RED_YELLOW ||
34                             status == AUTO_INIT))
35 {
36     num_to_display_1--;
37     num_to_display_2--;
38     led_buffer[0] = num_to_display_1/10;
39     led_buffer[1] = num_to_display_1%10;
40     led_buffer[2] = num_to_display_2/10;
41     led_buffer[3] = num_to_display_2%10;
42     set_timer(5, 1000);
43 }
44 if(is_timer_expired(4))
45 {
46     if(i == 4) i = 0;
47     update7SEG(i);
48     display7SEG(led_buffer[i]);
49     i++;
50     set_timer(4, 150);
51 }
52
53 if(is_timer_expired(2))
54 {
55     switch (status)
56     {
57         case AUTO_INIT:
58             status = AUTO_RED_GREEN;
59             break;
60         case AUTO_RED_GREEN:
61             led_red_and_green();
```

```
62     num_to_display_1 = 5;
63     num_to_display_2 = 3;
64     status = AUTO_RED_YELLOW;
65     set_timer(2, 3000);
66     break;
67 case AUTO_RED_YELLOW:
68     led_red_and_yellow();
69     num_to_display_1 = 2;
70     num_to_display_2 = 2;
71     status = AUTO_GREEN_RED;
72     set_timer(2, 2000);
73     break;
74 case AUTO_GREEN_RED:
75     led_green_and_red();
76     num_to_display_1 = 3;
77     num_to_display_2 = 5;
78     status = AUTO_YELLOW_RED;
79     set_timer(2, 3000);
80     break;
81 case AUTO_YELLOW_RED:
82     led_yellow_and_red();
83     num_to_display_1 = 2;
84     num_to_display_2 = 2;
85     status = AUTO_GREEN_RED;
86     set_timer(2, 2000);
87     break;
88 default:
89     break;
90 }
91 }
92 }
```

Listing 3: Sourcecode of fsm_auto.c

To add code for blinking LEDs depending on the mode that is selected.

```
1 #include "fsm_manual.h"
2
3 int red_time = 0;
4 int yellow_time = 0;
```

```
5 int green_time = 0;
6 int red;
7 int yellow;
8 int green;
9 int check = 0;
10 int num_to_display1 = 0;
11 int num_to_display2 = 0;
12
13 void fsm_manual()
14 {
15     if(is_timer_expired(8))
16     {
17         getKeyInput();
18         set_timer(8, 20);
19     }
20
21     if(is_timer_expired(9))
22     {
23         switch(status)
24         {
25             case MANUAL_INIT:
26                 toggle();
27                 led_buffer[0] = 0;
28                 led_buffer[1] = 0;
29                 led_buffer[2] = 0;
30                 led_buffer[3] = 0;
31                 if(isButton0Pressed())
32                 {
33                     clear();
34                     status = RED_SET;
35                 }
36                 else if(isButton2Pressed())
37                 {
38                     clear();
39                     status = AUTO_INIT;
40                 }
41                 else set_timer(3, 20);
42                 set_timer(9, 500);
43                 break;
```

```
44     case MAN_RED_GREEN:
45         led_red_and_green();
46         led_buffer[0] = red/10;
47         led_buffer[1] = red%10;
48         led_buffer[2] = green/10;
49         led_buffer[3] = green%10;
50         red--;
51         green--;
52         if(green == 0)
53         {
54             green = green_time;
55             status = MAN_RED_YELLOW;
56         }
57         if(isButton2Pressed())
58         {
59             clear();
60             red_time = 0; yellow_time = 0; green_time = 0;
61             status = MANUAL_INIT;
62             set_timer(3, 20);
63         }
64         set_timer(9, 1000);
65         break;
66     case MAN_RED_YELLOW:
67         led_red_and_yellow();
68         led_buffer[0] = red/10;
69         led_buffer[1] = red%10;
70         led_buffer[2] = yellow/10;
71         led_buffer[3] = yellow%10;
72         red--;
73         yellow--;
74         if(red == 0) red = red_time;
75         if(yellow == 0)
76         {
77             yellow = yellow_time;
78             status = MAN_GREEN_RED;
79         }
80         if(isButton2Pressed())
81         {
82             clear();
```

```
83     red_time = 0; yellow_time = 0; green_time = 0;
84     status = MANUAL_INIT;
85     set_timer(3, 20);
86 }
87 set_timer(9, 1000);
88 break;
89 case MAN_GREEN_RED:
90     led_green_and_red();
91     led_buffer[0] = green/10;
92     led_buffer[1] = green%10;
93     led_buffer[2] = red/10;
94     led_buffer[3] = red%10;
95     red--;
96     green--;
97     if(green == 0)
98     {
99         green = green_time;
100        status = MAN_YELLOW_RED;
101    }
102    if(isButton2Pressed())
103    {
104        clear();
105        red_time = 0; yellow_time = 0; green_time = 0;
106        status = MANUAL_INIT;
107        set_timer(3, 20);
108    }
109    set_timer(9, 1000);
110    break;
111 case MAN_YELLOW_RED:
112     led_yellow_and_red();
113     led_buffer[0] = yellow/10;
114     led_buffer[1] = yellow%10;
115     led_buffer[2] = red/10;
116     led_buffer[3] = red%10;
117     yellow--;
118     red--;
119     if(yellow == 0)
120     {
121         red = red_time;
```



```
122     yellow = yellow_time;
123     status = MAN_RED_GREEN;
124 }
125 if(isButton2Pressed())
126 {
127     clear();
128     red_time = 0; yellow_time = 0; green_time = 0;
129     status = MANUAL_INIT;
130     set_timer(3, 20);
131 }
132 set_timer(9, 1000);
133 break;
134 default:
135     break;
136 }
137 }
138 }
```

Listing 4: Source code of fsm_manual.c

7 Exercise 7: Adding code for increasing time duration value for the red LEDs, amber, greenLEDs

In this exercise, this source code I show is also used for Exercise 8, 9

```
1 #include "fsm_manual.h"
2
3 int red_time = 0;
4 int yellow_time = 0;
5 int green_time = 0;
6 int red;
7 int yellow;
8 int green;
9 int check = 0;
10
11 int num_to_display1 = 0;
12 int num_to_display2 = 0;
13
14 void fsm_manual()
15 {
```

```
16  if(is_timer_expired(8))
17  {
18      getKeyInput();
19      set_timer(8, 20);
20  }
21
22  if(is_timer_expired(9))
23  {
24      switch(status)
25      {
26          case MANUAL_INIT:
27              toggle();
28              led_buffer[0] = 0;
29              led_buffer[1] = 0;
30              led_buffer[2] = 0;
31              led_buffer[3] = 0;
32              if(isButton0Pressed())
33              {
34                  clear();
35                  status = RED_SET;
36              }
37              else if(isButton2Pressed())
38              {
39                  clear();
40                  status = AUTO_INIT;
41              }
42              else set_timer(3, 20);
43              set_timer(9, 500);
44              break;
45          case RED_SET:
46              togglered();
47              if(isButton0Pressed())
48              {
49                  clear();
50                  if (red_time <= 1) red_time = 2;
51                  else red = red_time;
52                  status = YELLOW_SET;
53                  led_buffer[0] = yellow_time/10;
54                  led_buffer[1] = yellow_time%10;
```

```
55     led_buffer[2] = 0;
56     led_buffer[3] = 0;
57     set_timer(3, 20);
58 }
59 else if(isButton1Pressed())
60 {
61     clear();
62     red_time++;
63     led_buffer[0] = red_time/10;
64     led_buffer[1] = red_time%10;
65     led_buffer[2] = 0;
66     led_buffer[3] = 0;
67     set_timer(3, 20);
68 }
69 else if(isButton2Pressed())
70 {
71     clear();
72     if(red_time <= 1)
73     {
74         red_time = 2;
75         yellow_time = 1;
76         green_time = red_time - yellow_time;
77
78         red = red_time;
79         yellow = yellow_time;
80         green = green_time;
81
82         status = MAN_RED_GREEN;
83     }
84     else
85     {
86         yellow_time = 1;
87         green_time = red_time - yellow_time;
88         red = red_time;
89         yellow = yellow_time;
90         green = green_time;
91         led_buffer[0] = red_time/10;
92         led_buffer[1] = red_time%10;
93         led_buffer[2] = 0;
```

```
94         led_buffer[3] = 0;
95         status = MAN_RED_GREEN;
96     }
97     set_timer(3, 20);
98 }
99 else set_timer(3, 20);
100 set_timer(9, 500);
101 break;
102 case YELLOW_SET:
103     toggleyellow();
104     if(isButton0Pressed())
105     {
106         clear();
107         if (yellow_time <= 1) yellow_time = 1;
108         status = GREEN_SET;
109         led_buffer[0] = green_time/10;
110         led_buffer[1] = green_time%10;
111         led_buffer[2] = 0;
112         led_buffer[3] = 0;
113         set_timer(3, 20);
114     }
115     else if(isButton1Pressed())
116     {
117         yellow_time++;
118         led_buffer[0] = yellow_time/10;
119         led_buffer[1] = yellow_time%10;
120         led_buffer[2] = 0;
121         led_buffer[3] = 0;
122         set_timer(3, 20);
123     }
124     else if(isButton2Pressed())
125     {
126         clear();
127         if(yellow_time < 1)
128         {
129             yellow_time = 1;
130             green_time = red_time - yellow_time;
131             red = red_time;
132             yellow = yellow_time;
```

```
133         green = green_time;
134         led_buffer[0] = yellow_time/10;
135         led_buffer[1] = yellow_time%10;
136         led_buffer[2] = 0;
137         led_buffer[3] = 0;
138         status = MAN_RED_GREEN;
139     }
140     else if (yellow_time >= red_time)
141     {
142         red_time = 0;
143         yellow_time = 0;
144         green_time = 0;
145         led_buffer[0] = 0;
146         led_buffer[1] = 0;
147         led_buffer[2] = 0;
148         led_buffer[3] = 0;
149         status = RED_SET;
150     }
151     else{
152         yellow = yellow_time;
153         green = green_time = red_time - yellow_time;
154         status = MAN_RED_GREEN;
155     }
156     set_timer(3, 20);
157 }
158 else set_timer(3, 20);
159 set_timer(9, 500);
160 break;
161 case GREEN_SET:
162     togglegreen();
163     if(isButton0Pressed())
164     {
165         clear();
166         status = RED_SET;
167         set_timer(3, 20);
168     }
169     else if(isButton1Pressed())
170     {
171         green_time++;
```

```
172     led_buffer[0] = green_time/10;
173     led_buffer[1] = green_time%10;
174     led_buffer[2] = 0;
175     led_buffer[3] = 0;
176     set_timer(3, 20);
177 }
178 else if(isButton2Pressed())
179 {
180     if(green_time == (red_time - yellow_time) && (yellow_time
!= 0))
181     {
182         clear();
183         status = MAN_RED_GREEN;
184     }
185     else {
186         status = RED_SET;
187         red_time = 0;
188         yellow_time = 0;
189         green_time = 0;
190         led_buffer[0] = 0;
191         led_buffer[1] = 0;
192         led_buffer[2] = 0;
193         led_buffer[3] = 0;
194     }
195     set_timer(3, 20);
196 }
197 else set_timer(3, 20);
198 set_timer(9, 500);
199 break;
200 default:
201     break;
202 }
203 }
204
205 }
```

Listing 5: Source code of fsm_manual.c



8 Link

For more details. please refer to this source code [Link Github](#)